

APPLICATION FOR UNITED STATES LETTERS PATENT

FOR

**A Stream Cipher Having  
A Shuffle Network Combiner Function**

**"Express Mail" mailing label number** EL 414969073 US  
**Date of Deposit** AUGUST 29, 1999  
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Address" twice on the date indicated above and that this paper or fee has been transmitted to the Assistant Commissioner for Patents, Washington, D.C. 20231  
Judith A. Namyko 8/29/99  
Signature Date

Inventor(s): **Gary L. Graunke  
David A. Lee  
Robert W. Faber**

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN, LLP  
12400 Wilshire Boulevard, 7th Floor  
Los Angeles, California 90025  
(503) 684-6200

"Express Mail" label number EL 41469073 US

**A Stream Cipher Having A Shuffle Network Combiner Function**

**BACKGROUND OF THE INVENTION**

5 1. Field of the Invention

The present invention relates to the field of cryptography. More specifically, the present invention relates to the robustness of stream ciphers.

2. Background Information

10 Cryptographic ciphers can be broadly divided into block ciphers and stream ciphers. Block ciphers cipher a block of plain text into ciphered text by applying multiple successive rounds of transformation to the plain text, using a cipher key. An example of a block cipher is the well known DES cipher. Stream ciphers cipher a stream of plain data into ciphered data by combining the stream of plain data with  
15 a pseudo random sequence dynamically generated using a cipher key. An example of a stream cipher is the well known XPF/KPD cipher.

Most stream ciphers employ one or more linear feedback shift registers (LFSR). In various applications, it is desirable to employ multiple LFSRs to increase the robustness of a stream cipher. However, employment of multiple LFSRs  
20 requires employment of a combiner function to recombine the multiple data bits output by the LFSRs. Most combiner functions known in the art are inefficient in their real estate requirement for hardware implementations. Thus, a robust stream cipher with a more efficient combiner function is desired.

25

A stream cipher is provided with one or more data bit generators to generate a first, second and third set of data bits. The stream cipher is further provided with a combiner function having a network of shuffle units to combine the third set of data bits, using the first and second sets of data bits as input data bits and control signals respectively of the network of shuffle units.

## BRIEF DESCRIPTION OF DRAWINGS

The present invention will be described by way of exemplary embodiments, but not limitations, illustrated in the accompanying drawings in which like references  
5 denote similar elements, and in which:

**Figure 1** illustrates an overview of the combined block/stream cipher of the present invention, in accordance with one embodiment;

**Figure 2** illustrates the block key section of **Fig. 1** in further detail, in accordance with one embodiment;

10 **Figure 3** illustrates the block data section of **Fig. 1** in further detail, in accordance with one embodiment;

**Figures 4a-4c** illustrate the stream data section of **Fig. 1** in further detail, in accordance with one embodiment; and

15 **Figure 5** illustrates a bit-wise view of the mapping section of **Fig. 1** in further detail, in accordance with one embodiment.

DETAILED DESCRIPTION OF THE INVENTION

In the following description, various aspects of the present invention will be described, and various details will be set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to those skilled in the art that the present invention may be practiced with only some or all aspects of the present invention, and the present invention may be practiced without the specific details. In other instances, well known features are omitted or simplified in order not to obscure the present invention.

Various operations will be described as multiple discrete steps performed in turn in a manner that is most helpful in understanding the present invention. However, the order of description should not be construed as to imply that these operations are necessarily performed in the order they are presented, or even order dependent. Lastly, repeated usage of the phrase "in one embodiment" does not necessarily refer to the same embodiment, although it may.

Referring now to **Figure 1**, wherein a block diagram illustrating the combined block/stream cipher of the present invention, in accordance with one embodiment, is shown. As illustrated, combined block/stream cipher **110** includes block key section **502**, data section **504**, stream key section **506**, and mapping section **508**, coupled to one another. Block key section **502** and data section **504** are employed in both the block mode as well as the stream mode of operation, whereas stream key section **506** and mapping section **508** are employed only in the stream mode of operation.

Briefly, in block mode, block key section **502** is provided with a block cipher key, such as an authentication key  $K_m$  or a session key  $K_s$  of a video content protection application; whereas data section **504** is provided with the plain text, such

as a basis random number  $A_n$  or a derived random number  $M_{i-1}$  of a video content protection application. "Rekeying enable" signal is set to a "disabled" state, operatively de-coupling block key section 502 from stream key section 506 during the block mode of operation.

Sub  
a's  
5 [A video content protection application that uses  $K_m$ ,  $K_x$ ,  $A_n$  and  $M_i$  is described in copending U.S. Patent Applications, serial numbers, <to be inserted>, filed contemporaneously, both entitled "Digital Video Content Transmission Ciphering/Deciphering Method and Apparatus", having common assignee and inventorship with the present application.]

10 During each clock cycle, the block cipher key as well as the plain text are transformed. The block cipher key is independently transformed, whereas transformation of the plain text is dependent on the transformation being performed on the block cipher key. After a desired number of clock cycles, the provided plain text is transformed into ciphered text. For the video content protection method  
15 disclosed in above mentioned co-pending applications, when block key section 502 is provided with  $K_m$  and data section 504 is provided with the  $A_n$ , ciphered  $A_n$  is read out and used as the session key  $K_s$ . When block key section 502 is provided with  $K_s$  and data section 504 is provided with the  $M_{i-1}$ , ciphered  $M_{i-1}$  is read out and used as the frame key  $K_i$ .

20 To decipher the ciphered plain text, block key section 502 and data section 504 are used in like manner as described above to generate the intermediate "keys", which are stored away (in storage locations not shown). The stored intermediate "keys" are then applied to the ciphered text in reversed order, resulting in the deciphering of the ciphered text back into the original plain text. Another approach  
25 to deciphering the ciphered text will be described after block key section 502 and

data section **504** have been further described in accordance with one embodiment each, referencing **Figs. 2-3**.

In stream mode, stream key section **506** is provided with a stream cipher key, such as a session key  $K_s$  or a frame key  $K_i$  of a video content protection application.

5 Block key section **502** and data section **504** are provided with random numbers, such as a session/frame keys  $K_s/K_i$  and a derived random numbers  $M_{i-1}$  of a video content protection application. "Rekeying enable" signal is set to an "enabled" state, operatively coupling block key section **502** to stream key section **506**. Periodically, at predetermined intervals, such as the horizontal blanking intervals of a video

10 frame, stream key section **506** is used to generate one or more data bits to dynamically modify the then current state of the random number stored in block data section **502**. During each clock cycle, in between the predetermined intervals, both random numbers stored in block key section **502** and data section **504** are transformed. The random number provided to block key section **502** is

15 independently transformed, whereas transformation of the random number provided to data section **504** is dependent on the transformation being performed in block key section **502**. Mapping block **506** retrieves a subset each, of the newly transformed states of the two random numbers, and reduces them to generate one bit of the pseudo random bit sequence. Thus, in a desired number of clock cycles, a pseudo

20 random bit sequence of a desired length is generated.

For the illustrated embodiment, by virtue of the employment of the "rekeying enable" signal, stream key section **506** may be left operating even during the block mode, as its outputs are effectively discarded by the "rekeying enable" signal (set in a "disabled" state).

25

**Figure 2** illustrates the block key section of **Fig. 1** in further detail, in accordance with one embodiment. As illustrated, block key section **502** includes registers **602a-602c**, substitution boxes **604**, and linear transformation unit **606**. In block mode, registers **602a-602c** are collectively initialized to a block cipher key, e.g. the earlier mentioned authentication key  $K_m$  or session key  $K_s$ . In stream mode, registers **602a-602c** are collectively initialized to a random number, e.g. the earlier mentioned session key  $K_s$  or frame key  $K_i$ . Each round, substitution boxes **604** and linear transformation unit **606** modify the content of registers **602a-602c**. More specifically, substitution boxes **604** receive the content of register **602a**, modify it, and then store the substituted content into register **602c**. Similarly, linear transformation unit **606** receives the content of registers **602b** and **602c**, linearly transforms them, and then correspondingly stores the linearly transformed content into registers **602a** and **602b**.

Substitution boxes **604** and linear transformation unit **606** may be implemented in a variety of ways in accordance with well known cryptographic principles. One specific implementation is given in more detail below after the description of **Fig. 3**.

**Figure 3** illustrates the block data section of **Fig. 1** in further detail, in accordance with one embodiment. For the illustrated embodiment, data section **504** is similarly constituted as block key section **502**, except linear transformation unit **706** also takes into consideration the content of register **602b**, when transforming the contents of registers **702b-702c**. In block mode, registers **702a-702c** are collectively initialized with the target plain text, e.g. earlier described random number  $M_{i-1}$  or derived random number  $M_{i-1}$ . In stream mode, registers **702a-702c** are collectively initialized with a random number. Each round, substitution boxes **704**



and linear transformation unit **706** modify the content of registers **702a-702c** as described earlier for block key section **502** except for the differences noted above.

Again, substitution boxes **604** and linear transformation unit **606** may be implemented in a variety of ways in accordance with well known cryptographic principles.

In one implementation for the above described embodiment, each register **602a, 602b, 602c, 702a, 702b, 702c** is 28-bit wide. [Whenever registers **602a-602c** or **702a-702cb** collectively initialized with a key value or random number less than 84 bits, the less than 84-bit number is initialized to the lower order bit positions with the higher order bit positions zero filled.] Additionally, each set of substitution boxes **604** or **704** are constituted with seven 4 input by 4 output substitution boxes. Each linear transformation unit **606** or **706** produces 56 output values by combining outputs from eight diffusion networks (each producing seven outputs). More specifically, the operation of substitution boxes **604/704** and linear transformation unit **606/706** are specified by the four tables to follow. For substitution boxes **604/704**, the  $l$ th input to box  $J$  is bit  $l*7+J$  of register **602a/702a**, and output  $l$  of box  $J$  goes to bit  $l*7+j$  of register **602c/702c**. [Bit 0 is the least significant bit.] For each diffusion network (linear transformation unit **606** as well as **706**), the inputs are generally labeled  $I0-I6$  and the outputs are labeled  $O0-O6$ . The extra inputs for each diffusion network of the linear transformation unit **706** is labeled  $K0-K6$ .

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SK0	8	14	5	9	3	0	12	6	1	11	15	2	4	7	10	13
SK1	1	6	4	15	8	3	11	5	10	0	9	12	7	13	14	2
SK2	13	11	8	6	7	4	2	15	1	12	14	0	10	3	9	5
SK3	0	14	11	7	12	3	2	13	15	4	8	1	9	10	5	6
SK4	12	7	15	8	11	14	1	4	6	10	3	5	0	9	13	2
SK5	1	12	7	2	8	3	4	14	11	5	0	15	13	6	10	9
SK6	10	7	6	1	0	14	3	13	12	9	11	2	15	5	4	8
SB0	12	9	3	0	11	5	13	6	2	4	14	7	8	15	1	10
SB1	3	8	14	1	5	2	11	13	10	4	9	7	6	15	12	0
SB2	7	4	1	10	11	13	14	3	12	15	6	0	2	8	9	5
SB3	6	3	1	4	10	12	15	2	5	14	11	8	9	7	0	13
SB4	3	6	15	12	4	1	9	2	5	8	10	7	11	13	0	14
SB5	11	14	6	8	5	2	12	7	1	4	15	3	10	13	9	0
SB6	1	11	7	4	2	5	12	9	13	6	8	15	14	0	3	10

Table I – Substitution performed by each of the seven constituting substitution boxes of substitution boxes **604/704**.

5

	Diffusion Network Logic Function
<b>O<sub>0</sub></b>	$K_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_5 \oplus I_6$
<b>O<sub>1</sub></b>	$K_1 \oplus I_0 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_5 \oplus I_6$
<b>O<sub>2</sub></b>	$K_2 \oplus I_0 \oplus I_1 \oplus I_3 \oplus I_4 \oplus I_5 \oplus I_6$
<b>O<sub>3</sub></b>	$K_3 \oplus I_0 \oplus I_1 \oplus I_2 \oplus I_4 \oplus I_5 \oplus I_6$
<b>O<sub>4</sub></b>	$K_4 \oplus I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_5 \oplus I_6$
<b>O<sub>5</sub></b>	$K_5 \oplus I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_6$
<b>O<sub>6</sub></b>	$K_6 \oplus I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_5 \oplus I_6$

Table II – Diffusion networks for linear transformation unit **606/706** (continued in Tables III & IV).

	K1	K2	K3	K4	K5	K6	K7	K8
I <sub>0</sub>	Kz0	Ky0	Ky4	Ky8	Ky12	Ky16	Ky20	Ky24
I <sub>1</sub>	Kz1	Ky1	Ky5	Ky9	Ky13	Ky17	Ky21	Ky25
I <sub>2</sub>	Kz2	Ky2	Ky6	Ky10	Ky14	Ky18	Ky22	Ky26
I <sub>3</sub>	Kz3	Ky3	Ky7	Ky11	Ky15	Ky19	Ky23	Ky27
I <sub>4</sub>	Kz4	Kz7	Kz10	Kz13	Kz16	Kz19	Kz22	Kz25
I <sub>5</sub>	Kz5	Kz8	Kz11	Kz14	Kz17	Kz20	Kz23	Kz26
I <sub>6</sub>	Kz6	Kz9	Kz12	Kz15	Kz18	Kz21	Kz24	Kz27
O <sub>0</sub>	Kx0	Ky0	Ky1	Ky2	Ky3	Kx7	Kx8	Kx9
O <sub>1</sub>	Kx1	Ky4	Ky5	Ky6	Ky7	Kx10	Kx11	Kx12
O <sub>2</sub>	Kx2	Ky8	Ky9	Ky10	Ky11	Kx13	Kx14	Kx15
O <sub>3</sub>	Kx3	Ky12	Ky13	Ky14	Ky15	Kx16	Kx17	Kx18
O <sub>4</sub>	Kx4	Ky16	Ky17	Ky18	Ky19	Kx19	Kx20	Kx21
O <sub>5</sub>	Kx5	Ky20	Ky21	Ky22	Ky23	Kx22	Kx23	Kx24
O <sub>6</sub>	Kx6	Ky24	Ky25	Ky26	Ky27	Kx25	Kx26	Kx27

Table III – Diffusion networks for linear transformation unit **606/706** (continued in Table IV).

	B1	B2	B3	B4	B5	B6	B7	B8
I <sub>0</sub>	Bz0	By0	By4	By8	By12	By16	By20	By24
I <sub>1</sub>	Bz1	By1	By5	By9	By13	By17	By21	By25
I <sub>2</sub>	Bz2	By2	By6	By10	By14	By18	By22	By26
I <sub>3</sub>	Bz3	By3	By7	By11	By15	By19	By23	By27
I <sub>4</sub>	Bz4	Bz7	Bz10	Bz13	Bz16	Bz19	Bz22	Bz25
I <sub>5</sub>	Bz5	Bz8	Bz11	Bz14	Bz17	Bz20	Bz23	Bz26
I <sub>6</sub>	Bz6	Bz9	Bz12	Bz15	Bz18	Bz21	Bz24	Bz27
K <sub>0</sub>	Ky0	–	–	–	–	Ky7	Ky14	Ky21
K <sub>1</sub>	Ky1	–	–	–	–	Ky8	Ky15	Ky22
K <sub>2</sub>	Ky2	–	–	–	–	Ky9	Ky16	Ky23
K <sub>3</sub>	Ky3	–	–	–	–	Ky10	Ky17	Ky24
K <sub>4</sub>	Ky4	–	–	–	–	Ky11	Ky18	Ky25
K <sub>5</sub>	Ky5	–	–	–	–	Ky12	Ky19	Ky26
K <sub>6</sub>	Ky6	–	–	–	–	Ky13	Ky20	Ky27
O <sub>0</sub>	Bx0	By0	By1	By2	By3	Bx7	Bx8	Bx9
O <sub>1</sub>	Bx1	By4	By5	By6	By7	Bx10	Bx11	Bx12
O <sub>2</sub>	Bx2	By8	By9	By10	By11	Bx13	Bx14	Bx15
O <sub>3</sub>	Bx3	By12	By13	By14	By15	Bx16	Bx17	Bx18
O <sub>4</sub>	Bx4	By16	By17	By18	By19	Bx19	Bx20	Bx21
O <sub>5</sub>	Bx5	By20	By21	By22	By23	Bx22	Bx23	Bx24
O <sub>6</sub>	Bx6	By24	By25	By26	By27	Bx25	Bx26	Bx27

Table IV – Diffusion networks for linear transformation unit **606/706** (continued from Table III).

- 5 Referring now back to **Fig. 5**, recall that a ciphered text may be deciphered by generating the intermediate “keys” and applying them backward. Alternatively, for an embodiment where either the inverse of substitution boxes **604/704** and linear transformation units **606/706** are included or they may be dynamically reconfigured to operate in an inverse manner, the ciphered text may be deciphered as follows.
- 10 First, the cipher key used to cipher the plain text is loaded into block key section **502**, and block key section **502** is advanced by R-1 rounds, i.e. one round short of

the number of rounds (R) applied to cipher the plain text. After the initial R-1 rounds, the ciphered text is loaded into data section 504, and both sections, block key section 502 and data section 504, are operated "backward", i.e. with substitution boxes 604/704 and linear transformation units 606/706 applying the inverse

5 substitutions and linear transformations respectively.

Figures 4a-4c illustrate the stream key section of Fig. 1 in further detail, in accordance with one embodiment. As illustrated in Fig. 4a, stream key section 506 includes a number of linear feedback shift registers (LFSRs) 802 and combiner function 804, coupled to each other as shown. LFSRs 802 are collectively initialized

10 with a stream cipher key, e.g. earlier described frame key Ki. During operation, the stream cipher key is successively shifted through LFSRs 802. Selective outputs are taken from LFSRs 802, and combiner function 804 is used to combine the selective outputs. In stream mode (under which, rekeying is enabled), the combined result is used to dynamically modify a then current state of a block cipher key in block key

15 section 502.

For the illustrated embodiment, four LFSRs of different lengths are employed. Three sets of outputs are taken from the four LFSRs. The polynomials represented by the LFSR and the bit positions of the three sets of LFSR outputs are given by the table to follows:

20

LFSR	Polynomial	Combining Function		
		Taps		
		0	1	2
3	$X^{17} + X^{15} + X^{11} + X^5 + 1$	6	12	17
2	$X^{16} + X^{15} + X^{12} + X^8 + X^7 + X^5 + 1$	6	10	16
1	$X^{14} + X^{11} + X^{10} + X^7 + X^6 + X^4 + 1$	5	9	14
0	$X^{13} + X^{11} + X^9 + X^5 + 1$	4	8	13

Table V – Polynomials of the LFSR and tap positions.

The combined result is generated from the third set of LFSR outputs, using the first and second set of LFSR outputs as data and control inputs respectively to combiner function 802. The third set of LFSR outputs are combined into a single bit. In stream mode (under which, rekeying is enabled), the combined single bit is then used to dynamically modify a predetermined bit of a then current state of a block cipher key in block key section 502.

Fig. 4b illustrates combiner function 804 in further detail, in accordance with one embodiment. As illustrated, combiner function 804 includes shuffle network 806 and XOR 808a-808b, serially coupled to each other and LFSRs 802 as shown. For the illustrated embodiment, shuffle network 806 includes four binary shuffle units 810a-810d serially coupled to each other, with first and last binary shuffle units 810a and 810d coupled to XOR 808a and 808b respectively. XOR 808a takes the first group of LFSR outputs and combined them as a single bit input for shuffle network 806. Binary shuffle units 810a-810d serially propagate and shuffle the output of XOR 808a. The second group of LFSR outputs are used to control the shuffling at corresponding ones of binary shuffle units 810a-810d. XOR 808b combines the third set of LFSR outputs with the output of last binary shuffle unit 810d.

**Fig. 4c** illustrates one binary shuffle unit **810\*** (where \* is one of **a-d**) in further detail, in accordance with one embodiment. Each binary shuffle unit **810\*** includes two flip-flops **812a** and **812b**, and a number of selectors **814a-814c**, coupled to each other as shown. Flip-flops **812a** and **812b** are used to store two state values (A, B). Each selector **814a**, **814b** or **814c** receives a corresponding one of the second group of LFSR outputs as its control signal. Selector **814a-814b** also each receives the output of XOR **808a** or an immediately preceding binary shuffle unit **810\*** as input. Selector **814a-814b** are coupled to flip-flops **812a-812b** to output one of the two stored state values and to shuffle as well as modify the stored values in accordance with the state of the select signal. More specifically, for the illustrated embodiment, if the stored state values are (A, B), and the input and select values are (D, S), binary shuffle unit **810\*** outputs A, and stores (B, D) if the value of S is "0". Binary shuffle unit **810\*** outputs B, and stores (D, A) if the value of S is "1".

Referring now to back to **Figure 1**, as illustrated and described earlier, mapping function **508** generates the pseudo random bit sequence based on the contents of selected registers of block key section **502** and data section **504**. In one embodiment, where block key section **502** and data section **504** are implemented in accordance with the respective embodiments illustrated in **Fig. 2-3**, mapping function **508** generates the pseudo random bit sequence at 24-bit per clock based on the contents of registers (Ky and Kz) **602b-602c** and (By and Bz) **702b-702c**. More specifically, each of the 24 bits is generated by performing the XOR operation on nine terms in accordance with the following formula:

$$(B0 \bullet K0) \oplus (B1 \bullet K1) \oplus (B2 \bullet K2) \oplus (B3 \bullet K3) \oplus (B4 \bullet K4) \oplus (B5 \bullet K5) \oplus (B6 \bullet K6) \oplus B7 \oplus K7$$

Where “ $\oplus$ ” represents a logical XOR function, “ $\bullet$ ” represents a logical AND function, and the input values B and K for the 24 output bits are

Input Origin Output bit	B0 Bz	B1 Bz	B2 Bz	B3 Bz	B4 Bz	B5 Bz	B6 Bz	B7 By	K0 Kz	K1 Kz	K2 Kz	K3 Kz	K4 Kz	K5 Kz	K6 Kz	K7 Ky
0	14	23	7	27	3	18	8	20	12	24	0	9	16	7	20	13
1	20	26	6	15	8	19	0	10	26	18	1	11	6	20	12	19
2	7	20	2	10	19	14	26	17	1	22	8	13	7	16	25	3
3	22	12	6	17	3	10	27	4	24	2	9	5	14	18	21	15
4	22	24	14	18	7	1	9	21	19	24	20	8	13	6	3	5
5	12	1	16	5	10	24	20	14	27	2	8	16	15	22	4	21
6	5	3	27	8	17	15	21	12	14	23	16	10	27	1	7	17
7	9	20	1	16	5	25	12	6	9	13	22	17	1	24	5	11
8	23	25	11	13	17	1	6	22	25	21	18	15	6	11	1	10
9	4	0	22	17	25	10	15	18	0	20	26	19	4	15	9	27
10	23	25	9	2	13	16	4	8	2	11	27	19	14	22	4	7
11	3	6	20	12	25	19	10	27	24	3	14	6	23	17	10	1
12	26	1	18	21	14	4	10	0	17	7	26	0	23	11	14	8
13	2	11	4	21	15	24	18	9	5	16	12	2	26	23	11	6
14	22	24	3	19	11	4	13	5	22	0	18	8	25	5	15	2
15	12	0	27	11	22	5	16	1	10	3	15	19	21	27	6	18
16	24	20	2	7	15	18	8	3	12	20	5	19	1	27	8	23
17	12	16	8	24	7	2	21	23	17	2	11	14	7	25	22	16
18	19	3	22	9	13	6	25	7	4	10	2	17	21	24	13	22
19	11	17	13	26	4	21	2	16	3	4	13	26	18	23	9	25
20	17	23	26	14	5	11	0	15	26	3	9	19	21	12	6	0
21	9	14	23	16	27	0	6	24	18	21	3	27	4	10	15	26
22	7	21	8	13	1	26	19	25	25	0	12	10	7	17	23	9
23	27	15	23	5	0	9	18	11	8	0	25	20	16	5	13	12

5 Accordingly, a novel dual use block or stream cipher has been described.

### Epilogue

From the foregoing description, those skilled in the art will recognize that many other variations of the present invention are possible. In particular, while the present invention has been described with the illustrated embodiments, non-LFSR based stream key section, more or less block key registers, larger or smaller block



5

---

[illegible]